



**CERTIFICATE OF MAILING**

I hereby certify that this correspondence is being deposited with the U.S. Postal Service for First-Class Mail Delivery, postage prepaid, addressed to: Mail Stop Appeal Brief – Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on 28 December 2005.

  
Lynne M. Milliot

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Application of:

Theodore Allen HUCK et al.

Application No. 09/256,845

Confirmation No. 1360

Filed: 24 February 1999

Title: **SYSTEM AND METHODOLOGY FOR  
DYNAMIC APPLICATION ENVIRONMENT  
EMPLOYING RUNTIME EXECUTION  
TEMPLATES**

Group Art Unit: 2176

Examiner: Almari Romero YUAN

**CUSTOMER NO. 22470**

**MAIL STOP APPEAL BRIEF - PATENTS**

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

**APPEAL BRIEF**

Sir:

This Appeal Brief is filed in support of Appellants' appeal from the Office Action, mailed 27 May 2005, in this case. A Notice of Appeal was filed on 29 August 2005. An appropriate request for an extension of time accompanies this paper.

The appropriate fee as set forth in § 41.20 (b)(2) of \$500.00 is covered in the enclosed check. Should it be determined that additional fees are required, the Commissioner is hereby authorized to charge those fees to Deposit Account No. 50-0869 (Attorney Docket No. PUMA 1000-1).

**TABLE OF CONTENTS**

I. REAL PARTY IN INTEREST .....	1
II. RELATED APPEALS AND INTERFERENCES .....	1
III. STATUS OF CLAIMS .....	1
IV. STATUS OF AMENDMENTS .....	1
V. SUMMARY OF CLAIMED SUBJECT MATTER .....	1
VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL .....	2
VII. ARGUMENT .....	3
A. Preliminary Review of the Technology Disclosed and References .....	3
1. The Disclosed Technology .....	3
2. The Krishna Reference .....	4
3. The Kiyono Reference .....	5
4. The Foley Reference .....	6
B. Rejection of Independent Claim 60 Under Section 103(a) as Unpatentable over Krishna in view of Kiyono, in further view of Foley, was Improper .....	6
1. Abstract References .....	6
2. Dictionary .....	8
C. Rejection of Dependent Claims 10-12, 14, 23-25, 30-34, 36-38, 40-43 and 45 as Unpatentable over Krishna in view of Kiyono, in further view of Foley, was Improper .....	9
D. Rejection of Independent Claim 46 Under Section 103(a) as Unpatentable over Krishna in view of Foley was Improper .....	10
E. Rejection of Dependent Claims 2-6, 19-20, 22, and 47-59 as Unpatentable over Krishna in view of Foley, was Improper .....	11
CONCLUSION .....	12
CLAIMS APPENDIX .....	13

### **I. REAL PARTY IN INTEREST**

The real party in interest is Intellisync Corporation, the assignee of record (recordation pending). On 14 November 2005, a Patent Assignment was submitted to the Patent Office by assignee transferring ownership of the referenced application from Starfish Software, Inc. to Intellisync Corporation. A Notice of Recordation has not yet been received from the Patent Office.

### **II. RELATED APPEALS AND INTERFERENCES**

There are no known appeals or interferences relating to this case.

### **III. STATUS OF CLAIMS**

Claims 2-6, 10-12, 14, 19, 20, 22-25, 30-34, 36-38, 40-43 and 45-60 are pending in this application. All have been rejected and all of the rejections are subject to this appeal.

### **IV. STATUS OF AMENDMENTS**

The original claims of this application were claims 1 through 60. A response to the Office Action mailed 21 October 2004 was filed on 21 March 2005, together with a Request for Continued Examination, canceling claims 1, 7-9, 13, 15-18, 21, 26-29, 35, 39 and 44, and leaving claims 2-6, 10-12, 14, 19, 20, 22-25, 30-34, 36-38, 40-43 and 45-60 to be examined in this application.

In response to the most recent Office Action mailed 27 May 2005, Applicants filed a Notice of Appeal on 29 August 2005.

No amendment has been filed after the Office Action mailed 27 May 2005.

### **V. SUMMARY OF CLAIMED SUBJECT MATTER**

There are two independent claims, numbers 46 and 60, which are addressed individually.

Claim 46 presents a method of creating and deploying an application that provides access to back-end information access functionality. The method includes three phases. First, creating at least one template including one or more abstract references that specify functionality to be invoked when a given client requests the template. Appln. p. 18, line 1 – p. 22, line 23. Second, registering the abstract

references with a dictionary that associates the abstract references with at least one run-time handler and one or more run-time services. Appln. p. 22, line 23 -- p. 23, line 2. And third, providing the specified functionality to access information, including receiving a request from the given client that identifies at least one template; accessing the identified template and determining the abstract references in the identified template; accessing the dictionary and determining the run-time handler and the run-time services associated with the abstract references; and invoking the run-time handler and the run time services to access to the back-end information access functionality. Appln. p. 23, lines 3 --23.

Claim 60 describes a template repository and template manager system that provide access to a back-end information access functionality in response to a client request for a template. This systems comprises the following elements: a template repository that stores templates, a particular template including one or more abstract references that specify back-end information access functionality to be invoked when the client requests the particular template (Appln. p. 10, lines 3 –24); a dictionary that associates the abstract references with one or more run-time services (Appln. p. 12, lines 9 – 16); and a template manager, responsive to the client request that identifies the particular template including logic that accesses the template repository and parses the particular template, accesses the dictionary and resolves the abstract references to the associated run-time services, and invokes the associated run-time services that provide back-end information access functionality (Appln. p. 11, line 3 – p. 12, line 23).

## **VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL**

First, whether it was improper to reject independent claim 60 and dependent claims 10-12, 14, 23-25, 30-34, 36-38, 40-43 and 45 under 35 U.S.C. 103(a) as being unpatentable over Krishna, in view of Kiyono, and further in view of Foley?

Second, whether it was improper to reject independent claim 46 and dependent claims 2-6, 19-20, 22 and 47-59 under 35 U.S.C. 103(a) as being unpatentable over Krishna, in view of Foley?

## **VII. ARGUMENT**

### **A. Preliminary Review of the Technology Disclosed and References**

#### **1. The Disclosed Technology**

The technology of the present application allows one to deal with constant changes in the code base of a software package. Such changes arise from the need to support a number of different hardware or operating system platforms, or from the need for system upgrades, whether minor (such as bug fixes) or major (functionality additions, for example). An aspect of the present application distinguishes between “hard” elements of the system, which do not change with either platform or version, and “soft” elements that can be easily modified for different versions or platforms. The “hard” portions of the system provide the ability to perform operations – parsing a document, executing commands, and so forth, while the “soft” portions provide the content and features, which can differ based on platform (providing different database access commands for different back-end database systems, for example) and functionality (such as the integration with specific software, such as Microsoft Windows).

As outlined in Fig. 3, aspects of the disclosed technology are templates, stored in a template repository 330, containing content. The content most conveniently is presented in HTML or a variant. A template is parsed by a template manager 340. It is often the case that pure HTML does not provide desired functionality, and therefore the system must call additional functions, which depend on the hardware or software platform in use. Database commands, for example, will differ based on the back-end database in use. Where such additional functionality is needed, the template includes a parameter. The template manager looks to the parameter dictionary 350, which provides information to the template manager which allows the latter element to call the appropriate service to perform the desired action. Application at 12, lines 1 – 14.

It can be readily seen that a combination of templates and the parameter dictionary allows straightforward version and platform control. The difference between a Macintosh version and a Windows version of software can be reduced to changes in the parameter dictionary, based on which set of services should be invoked. In the same manner, a change in functionality is accomplished by updating the services

themselves, together with the parameter dictionary, and (where required) the templates. Application at 12, lines 23 – 29.

It is important to focus on the nature of the parameters themselves. An aspect of the application is that a given parameter invokes different services, depending on the platform and version installed. Thus, parameters cannot be directly linked to services, either through direct system calls or devices such as pointers. Rather, parameters must be completely abstract, with their meaning provided only at the parameter dictionary.

Many modern software applications follow the client-server model. That is, functionality is divided between the client – the user, typically operating on a personal computer or workstation – and the server, typically a network server of either a general or special purpose (e.g., database server) type. Details may differ between, for example thin client vs. thick client variants, but the essence of this model is that functionality is split. In that regard, it should be noted that the present application sets out a client-based problem and a client-based solution. The problem at hand arises because clients differ, in both platform and version. The proffered solution is likewise client-based, with templates and the parameter dictionary, at a minimum, being located at the client. That importance of this point will become clear as the discussion proceeds.

## **2. The Krishna Reference**

Krishna addresses a completely different problem from that considered in the present application. As its title implies, the subject matter of this patent is a publishing system, in which a commercially-available software package known as Page Builder runs on a server to generate HTML pages which are then sent to clients. A somewhat detailed overview of this system is useful:

Briefly, Krishna's invention is a server-based Page Builder program that operates on a document template file to automatically produce output files which are downloaded to clients. The files take the form of either an output file that requires a downloadable viewer program, or a visual representation of each possible object and each navigable state of the template file, requiring a succession of client download operations. The latter case requires a set of links which specify how transitions between navigable state representations should occur.

More specifically, Krishna teaches a server-side system, with template files located on the server. The system uses the template files to prepare HTML files for the user, either as files that require a downloadable viewer program (Col 5, lines 12 – 15) to process screen actions, or the server generates separate HTML files for each possible visual state specified by the template, including transition conditions, so that the server can send subsequent screens as appropriate. Col. 5, lines 31 – 46.

Krishna's templates are based on regions defined on the desired web page:

The author specifies a set of instructions for obtaining and formatting content objects to be displayed in each region, as well as actions to be taken on or by such objects. These object and action definitions are then placed in a template file 112. It should be understood that the content portion of a template file 112 may actually be a reference to objects that may be stored on as server 122 that is located remotes to the Web server 102 on which the template file 112 is stored.

Col. 5, lines 3 – 11.

If the user has installed the downloadable viewer program, the template is downloaded to the client system and the viewer, in conjunction with the local Java interpreter, displays the web page. "The Viewer 136 will typically use URL's contained within the template file 112 to obtain the ontent and action information from servers 102 and/or 122. The content and action to information in the template file 112 are then used to change the visual display of the page in response to the user inputs." Col. 5, lines 25 – 30. These references are concrete – that is, references specify exactly where the system can access the desired functionality, in terms of a pointer or URL. An abstract reference, in contrast, would be in the form of a variable that requires interpretation.

Krishna clearly uses templates, but those templates do not contain abstract references. There is no template manager. There is no parameter dictionary.

### **3. The Kiyono Reference**

Kiyono is concerned with the problem of "repetitively preparing relatively highly regularity [sic] content such as an electronic catalog and an electronic teaching material [sic]." Col. 1, lines 64 – 66. The example presented in the specification addresses the task of preparing multimedia materials that contain text and illustrations about a number

of cities. The solution employs a static template, shown in Fig. 4, which includes a logic (data) structure, a layout structure, and physical operation structure showing operations common to the task at hand. In the example, operations are “common to multimedia catalogs of major world cities.” Col. 5, lines 12 – 30. When the program calls a particular template, the template operating means uses the data structure, layout structure and physical operation instructions to display appropriate information on a screen. Col. 5, lines 31 – 50. Differences in format, layout or function can be presented, within the overall situation of dealing with repetitive instances of the same types of information. See Figs. 7, 8.

Kiyono presents a solution to the problem of displaying many instances of the same kinds of data, particularly adapted to catalogs. The templates employed here contain no abstract references, nor is there a dictionary for resolving such references.

#### **4. The Foley Reference**

Foley allows a user to manage, create, edit, debug and compile software portfolios that can include several different types of components or projects. These components can be located remotely from the user, and components may contain references (typically URL's) to applets, which can be run when the document is downloaded or on user command. Foley does not employ templates, nor is there any reference to (or means to process) abstract references. Neither the word “template” nor the word “parameter” nor the word “dictionary” appears in this patent.

#### **B. Rejection of Independent Claim 60 Under Section 103(a) as Unpatentable over Krishna in view of Kiyono, in further view of Foley, was Improper**

The Examiner's rejection of claim 60 as unpatentable over Krishna in view of Kiyono in further view of Foley, fails on either of two grounds. First, that analysis elides the fact that none of the references teaches or discloses the use of abstract references. In addition, the Examiner mischaracterizes Foley as disclosing a dictionary. These arguments are set out separately, as follows.

##### **1. Abstract References**

This rejection fails to appreciate the meaning of the word “abstract.” An abstract symbol is one “whose meaning and use have not been determined by a general agreement but have to be defined for each use of the symbol.” The Authoritative



Dictionary of IEEE Standards Terms (IEEE Press, 2000) at 4. An “abstract reference” is thus a reference that requires further definition, as in a dictionary.

The Examiner does not recognize that an “abstract reference” differs from any reference, as she simply says that Krishna’s template includes:

the abstract references with one or more run-time services (web page may include address pointers (references) to other servers where information can be found; the template file consists of a set of objects ... actions taken on or by such objects; template file 112 may actually be a reference to objects that may be stored on a server 122 that is located remotely to the web server 102 on which the template file is stored; col. 1, lines 21-32 and line 63 – col. 2, line 21, col. 5, lines 3-15.

OA at 3.

Address pointers and URLs provide locations where information is to be found. Such references are concrete, because they set out actual, physical location. An abstract reference provides neither the information itself nor a location, but rather it constitutes a token requiring resolution. See Appln. P. 21, lines 21 – 30. One can simply call an address or URL and gain immediate access to information, while an abstract reference requires a means for resolving the reference into a direct or indirect reference where the information can be found.

The difference between abstract and concrete references lies at the heart of the present application, because it permits a template to be used in multiple platforms or versions. To take a simple example, a system call on a Macintosh platform will have a different address from a similar call on a Windows system. Under the Applicant’s disclosure, one can use the same template on the two platforms, employing different parameter dictionaries. Nothing in the cited art permits that functionality.

That concept is discussed fully in the application:

The compiled templates are processed by the Template Manager 421 (from the Abstract Presentation Template Services 340), which operates in conjunction with a Parameter Dictionary 423. As the Template Manager 421 reads each compiled HTML template object, the Parameter Dictionary 423 allows the Template Manager to determine which run-time services is to be invoked. **In effect, the Parameter Dictionary, which functions in a manner akin to a compiler’s symbol table, allows the Template Manager to resolve the various references pertinent to the compiled tokens which comprise an HTML object into one or more run-time service invocations.**

Appln, page 12, lines 7 – 14 (emphasis added).

The concept is claimed clearly as well. Claim 60 recites “one or more abstract references that specify back-end information access,” which references are associated by the dictionary “with one or more run-time services”.

Thus, the cited combination does not include a claim element, abstract references, rendering the rejection on that ground improper.

## **2. Dictionary**

Although claim 60 recites “a dictionary that associates the abstract references with one or more run-time services,” rather than locating a dictionary in the cited art, the Examiner focuses on the function of resolving abstract references, saying that Krishna:

resolves abstract references to the associated run-time services (the template file specifies how a web page is to be displayed; the server 102 has stored a downloadable Viewer file which contains applet or plug-in portion that contains programs used by the browser; the browser ascertains that the downloaded template file requires an applet and uses the applet conjunction with the local Java interpreter to display the web page; col. 1, line 63 – col. 2, line 8, lines 62-65 and col. 5, lines 3-46);

OA at 3.

Two points are immediately apparent. First, a rejection cannot simply ignore a claim element. The claim recites a “dictionary”, not just some means for resolving references. As noted above, the application is clear what is meant by “dictionary”, yet the Examiner attempts to ignore the point altogether. She should not be allowed to substitute unspoken assumption for analysis.

In any event, the Examiner is technically wrong in equating Krishna’s viewer program with a dictionary. As set out above, Krishna is clear that the viewer program simply calls URL references. That is not the way a dictionary operates, either in general or in the context of the present application. Indeed, this point is made clear by the claim language calling for “a dictionary that associates the abstract references with one or more run-time services.” The process of “associating” a reference with a service clearly requires the action of consulting some source that ties the reference and the

service together. Krishna never performs that step because he never needs it – the templates all reside on the server, and the problem addressed and solved by the Applicants never arises for him.

Having once addressed this element, however, the Examiner adds the following passage, almost as an afterthought:

However, Krishna and Kiyono do not explicitly disclose “a dictionary that associates the abstract references with one or more run-time services” and “accesses the dictionary and resolves abstract references to the associated run-time services”.

Foley on col. 5, lines 23-27 and lines 58-63 teaches executing all of the referenced files that are executable (i.e., the applets), in other words, referenced applets are stored in a remote system are to be automatically pulled and executed; see Figure 1 shows JWS Applets 140A contains a list (dictionary) of referenced stored applets 140A1-140A2 wherein the these referenced applets are pulled (accessed) to be executed.

OA at 4.

That paragraph changes the underpinning of this rejection. The question whether Foley teaches a dictionary is dealt with at length below, and that discussion is relied upon here as well for the proposition that Foley’s “list” is not a dictionary, and in fact the “list” is not even a list.

At bottom, the only source for a dictionary is contained in the present application. When Krishna and Kiyono and Foley are combined, nowhere in any of those disclosures does one find an element that can accept an abstract reference and associate that reference with a run-time service. It is therefore respectfully requested that this rejection should be reversed.

**C. Rejection of Dependent Claims 10-12, 14, 23-25, 30-34, 36-38, 40-43 and 45 as Unpatentable over Krishna in view of Kiyono, in further view of Foley, was Improper**

The claims dependent from Claim 60 stand or fall with that claim. As set out above, independent claim 60 is allowable. Therefore, the claims depending from claim 60 are similarly allowable. It is not necessary to review the specific rejections of those

claims, as they all are made based on the art discussed above. It is therefore respectfully suggested that these claims are in condition for allowance.

**D. Rejection of Independent Claim 46 Under Section 103(a) as Unpatentable over Krishna in view of Foley was Improper**

Independent claim 46, together with the claims dependent from it (Claims 2-6, 19-20, 22, and 47-59), were rejected under 35 USC 103(a) as being obvious in light of Krishna in view of Foley. Here, the Examiner starts by simply ignoring the dictionary (p. 10 – 11), but then equates Foley's supposed "list" of applets to a dictionary (p. 12). The Examiner summarized his position in responding to Applicant's previous response:

Referring to claim 60 which cites the argued limitation, Foley does teach a dictionary associating abstract references in templates with one or more run-time services, see Figure 1 shows JWS Applets 140A contains a list of referenced stored applets 140A1-140A2 wherein the these referenced applets are pulled to be executed. Furthermore, Foley on col. 5, lines 23-27 and lines 58-63 teaches when the browser retrieves the web document linked to a selected icon, it automatically will pull in and begin executing referenced applets, in other words, stored referenced applets that are associated with a document will be pulled and executed.

Rejection at 15.

That position mischaracterizes the Foley patent. In fact, not only does the suggested equation of a "list" with a "dictionary" fail, but no list actually is found. The cited "list of referenced stored applets" simply does not exist except in the drawing. A review of Foley's discussion of applets 140A is instructive:

The JWS toolbar specification 112A is composed of four **sub-groups of files**: icon specifications 114A, web documents 120A, **JWS applets 140A** and other referenced files 1498A. The elements 114A, 120A, 140A, 148A specify the appearance and, more importantly, the operation of a set of icons (Ai) 162Ai that are displayed on the display 108A as elements of a JWS program 150A in the course of project and/or portfolio management. Col. 4, ll. 34-46.

Each Web document 120A (which could have initially been sorted locally or remotely before being loaded into the memory 106A)

includes two elements: a title 122Ai and a set of references to its components 124Ai.

... [A] **reference 124Ai can be to a Java applet 140Ai** that is responsible for handling the operations associated with the icon 1Ai whose related Web document referenced the applet 140Ai. In this situation, when the JWS browser 154A retrieves the web document 120Ai linked to a selected icon 1Ai, it automatically will pull in and begin executing the referenced applet 140Ai (which could have been stored on a remote system). **The applet 140Ai, running in the JWS browser's virtual machine, can then implement the icon's associated operations** without needing to worry about network and operating system complexities, which are handled respectively by the local operating system and the JWS browser 154A. Col 5, ll. 9 – 32.

[In Fig 2] The JWS browser 154A then loads all of the files referenced in the document 120A1 and also executes any of the reference files that are executable(i.e., the applets). In this example, it is assumed that there is one **referenced executable**, the applet 140A1. Col. 5, ll. 58 – 63.

Moreover, other applets are referenced by screen icons (e.g., portfolio manager applet 140A1) and can be selected by the user. See col. 6, lines 1 – 36.

There are, therefore, direct references to applets both in the toolbar specification and in the loaded documents, and those references are used to call and run the applets, but there is no list of applets, and certainly no file that allows the system to employ an abstract reference system. Foley certainly does not teach a dictionary, and in fact she does not even teach a list. As was previously the case, a claim element is not present in or suggested by the cited art, singly or in combination.

The suggested combination of Krishna, Kiyono and Foley fails to teach the claimed invention, and thus it is respectfully suggested that this rejection should be withdrawn.

**E. Rejection of Dependent Claims 2-6, 19-20, 22, and 47-59 as Unpatentable over Krishna in view of Foley, was Improper**

As set out above, independent claim 46 is allowable. Therefore, the claims depending from claim 60 are similarly allowable. It is not necessary to review the specific rejections of those claims, as they all are made based on the art discussed

above. It is therefore respectfully suggested that these claims are in condition for allowance.


**CONCLUSION**

In view of the foregoing, Applicant/Appellant asks that this honorable Board reverse the Examiner's rejections of the claims. In addition, it is submitted that all claims are now allowable, and a notice of intent to issue a patent is respectfully requested.

The Commissioner is hereby authorized to charge any fee determined to be due in connection with this communication, or credit any overpayment, to our Deposit Account No. 50-0869 (Attorney Docket No. PUMA 1000-1).

Respectfully submitted,

Dated: 28 December 2005

  
\_\_\_\_\_  
Joseph E. Root  
Registration No. 30,678  
Attorney for Patent Owner

HAYNES BEFFEL & WOLFELD LLP

P.O. Box 366  
751 Kelly Street  
Half Moon Bay, CA 94019  
Telephone: 650.712.0340  
Facsimile: 650.712.0263

### **CLAIMS APPENDIX**

1. (Cancelled)
2. (Previously presented) The method of claim 46, wherein the at least one template comprises at least one page description language template.
3. (Previously presented) The method of claim 2, wherein said at least one page description language template comprises at least one Hypertext Markup Language (HTML) document.
4. (Previously presented) The method of claim 2, wherein said at least one page description language template comprises at least one Standard Generalized Markup Language (SGML) document.
5. (Previously presented) The method of claim 46, wherein the references are embedded in the at least one template using user-defined tags.
6. (Previously presented) The method of claim 46, wherein the back-end information access functionality that is actually invoked is determined based, at least in part, on which platform a given client executes.
7. - 9. (Cancelled)
10. (Previously presented) The method of claim 46, wherein a template manager stores parsed versions of templates in a template cache, so that each template need only be parsed once.
11. (Previously presented) The method of claim 10, wherein the parsed versions of templates are maintained on a persistent storage, so that the parsed templates are available from one session to another.

12. (Previously presented) The method of claim 10, wherein at least one parsed version of a template is flushed, so that said system is forced to again parse the at least one template.

13. (Cancelled)

14. (Previously presented) The method of claim 47, wherein the back-end database includes an SQL database system that retrieves information in response to SQL queries.

15. - 18. (Cancelled)

19. (Previously presented) The method of claim 46, wherein the back-end information access functionality invoked is based, at least in part, on a specific client session that is executing.

20. (Previously presented) The method of claim 46, wherein logic that implements the action of providing the specified functionality to access information consists of a single code base application that is deployed on multiple platforms.

21. (Cancelled)

22. (Previously presented) The method of claim 46, wherein the abstract references include tokens specifying programming constructs.

23. (Original) The method of claim 22, wherein said programming constructs include conditional logic statements.

24. (Original) The method of claim 23, wherein said conditional logic statements include "if" statements.



25. (Original) The method of claim 23, wherein said conditional logic statements include "for" loops.

26. - 29. (Cancelled)

30. (Previously presented) The system of claim 60, wherein the particular template includes at least one page description language template.

31. (Original) The system of claim 30, wherein said at least one page description language template comprises a Hypertext Markup Language (HTML) document.

32. (Original) The system of claim 30, wherein said at least one page description language template comprises a Standard Generalized Markup Language (SGML) document.

33. (Previously presented) The system of claim 60, wherein the abstract references are embedded in the particular template using user-defined tags.

34. (Previously presented) The system of claim 60, wherein which run-time services that are actually invoked is determined based, at least in part, on which platform a given client executes.

35. (Cancelled)

36. (Previously presented) The system of claim 60, wherein said template manager stores parsed templates in a template cache, so that each template need only be parsed once.

37. (Previously presented) The system of claim 36, wherein said parsed templates are maintained on a persistent storage, so that the parsed templates are available from one application execution session to another.

38. (Previously presented) The system of claim 36, wherein any parsed templates are occasionally flushed, so that said system is forced to again parse the particular template.

39. (Cancelled)

40. (Previously presented) The system of claim 60, wherein the back-end database comprises an SQL database system that retrieves information in response to SQL queries.

41. (Previously presented) The system of claim 60, wherein the particular template comprises at least one read-only template.

42. (Previously presented) The system of claim 60, wherein the particular template is loaded by browser software running at said particular client.

43. (Previously presented) The system of claim 60, wherein the particular template comprises an input form having a platform-specific presentation when rendered at a given client.

44. (Cancelled)

45. (Previously presented) The system of claim 60, wherein which run-time services are invoked is determined based, at least in part, on a specific client session that is executing.

46. (Previously presented) A method of creating and deploying an application that provides access to back-end information access functionality, including:  
creating at least one template including one or more abstract references that specify functionality to be invoked when a given client requests the template;  
registering the abstract references with a dictionary that associates the abstract references with at least one run-time handler and one or more run-time services;

and

providing the specified functionality to access information, including:

- receiving a request from the given client that identifies at least one template;
- accessing the identified template and determining the abstract references in the identified template;
- accessing the dictionary and determining the run-time handler and the run-time services associated with the abstract references; and
- invoking the run-time handler and the run time services to access to the back-end information access functionality.

47. (Previously presented) The method of claim 46, wherein the back-end functionality includes accessing information in a back-end database.

48. (Previously presented) The method of claim 46, wherein the back-end functionality includes accessing information in a configuration table.

49. (Previously presented) The method of claim 46, wherein the back-end functionality includes accessing information from machine services.

50. (Previously presented) The method of claim 46, wherein the request can be resolved to the given client, further including invoking the run-time handler and the run time services using parameters corresponding to the client identity.

51. (Previously presented) The method of claim 46, wherein the request can be resolved to a platform from which the request originates, further including invoking the run-time handler and the run time services using parameters corresponding to the platform.

52. (Previously presented) The method of claim 46, wherein the request can be resolved to the given client and a platform from which the request originates, further including invoking the run-time handler and the run time services using parameters corresponding to the given client and the platform.

53. (Previously presented) The method of claim 47, wherein the request can be resolved to the given client and a platform from which the request originates, further including invoking the run-time handler and the run time services using parameters corresponding to the given client and the platform.

54. (Previously presented) The method of claim 46, wherein the abstract references specify functionality that is independent of a platform from which the request originates.

55. (Previously presented) The method of claim 46, wherein the abstract references specify functionality that is independent of the given client.

56. (Previously presented) The method of claim 46, wherein the abstract references specify functionality that is independent of any particular back-end database.

57. (Previously presented) The method of claim 46, wherein the back-end database includes a synchronization engine.

58. (Previously presented) The method of claim 46, wherein the request can be resolved to a platform from which the request originates, further including composing a presentation adapted to the platform.

59. (Previously presented) The method of claim 58, wherein the run time services that access information in the back-end database are independent of logic that composes the adapted presentation.

60. (Previously presented) A template repository and template manager system that provide access to a back-end information access functionality in response to a client request for a template, including:

a template repository that stores templates, a particular template including one or more abstract references that specify back-end information access functionality to be invoked when the client requests the particular template;

a dictionary that associates the abstract references with one or more run-time services; and

a template manager, responsive to the client request that identifies the particular template, including logic that

accesses the template repository and parses the particular template,

accesses the dictionary and resolves the abstract references to the associated run-time services, and

invokes the associated run-time services that provide back-end information access functionality.